AUTOMATIC GOAL GENERATION FOR REINFORCEMENT LEARNING AGENTS
GIACOMO CARFI' – 520951

INTRODUCTION

TRADITIONAL REINFORCEMENT LEARNING ALGS.

In traditional reinforcement learning algorithms, the agent is trained in optimizing a **single** cost function in order to perform a **single** task.

However, there are many scenarios where an agent must be able to perform multiple tasks, such as navigating to varying positions in a room or moving objects to varying locations.

MULTI-TASK POLICY SEARCH

This method allows an agent to automatically discover the range of tasks that it is capable of performing in its environment.

The sequence of tasks provided to the agent is called **curriculum**.

The problem of **maximizing the average success rate** of the agent over all possible goals is considered. Success is defined as the probability of successfully reaching each goal by the current policy.

In order to efficiently maximize this objective, the algorithm must intelligently choose which goals to focus on at every training stage: goals should be at the appropriate level of difficulty for the current policy.

PROBLEM DEFINITION

Given a set of goals G, a range of reward functions indexed by a goal is considered. Each goal corresponds to a set of terminal states such that goal g is considered to be achieved when the agent is in one of these states.

$$G\ set\ of\ goals\quad \forall g \in G\ \exists\ S^g \subset S\ set\ of\ terminal\ states$$

$$reward\ function\ that\ returns\ 1\ if\ the\ agent\ has\ reached\ the\ goal$$

$$r^g(s_t, a_t, s_{t+1}) = \mathbb{1}\{s_{t+1} \in S^g\}$$

The objective of the agent is to learn a policy π that, given any goal g ∈ G, acts optimally with respect to r^g.

The expected return obtained when an action sampled from the policy is used can be expressed as the probability of success on that goal within T time-steps.

$$\begin{aligned} R^g(\pi) &= \mathbb{E}_{\pi(\cdot\,|\,s_t,g)}\, \mathbb{1}\{\exists\, t \in [1 \ldots T] : s_t \in S^g\} \\ &= \mathbb{P}\Big(\exists\, t \in [1 \ldots T] : s_t \in S^g \,\Big|\, \pi, g\Big) \end{aligned}$$

PROBLEM DEFINITION

The objective is to find a policy that achieves a high reward for many goals.

$$\pi^*(a_t \mid s_t, g) = \arg\max_{\pi} \mathbb{E}_{g \sim p_g(\cdot)} R^g(\pi)$$

$$p_g(g)\ distribution\ of\ goals$$

The objective measures the average probability of success over all goals sampled from the distribution of goal.

This is defined as **coverage.**

METHOD

Goal Labeling

Label a set of goals based on whether they are at the appropriate level of difficulty for the current policy.

The distribution from which goals are sampled during training is uniform over the set of the **Goals of Intermediate Difficulty.**

$$GOID_i \coloneqq \{g \colon R_{min} \leq R^g(\pi_i) \leq R_{max}\} \subseteq G$$

- $R^g(\pi_i) > R_{min}$ policy at timestep I need to receive some minimum expected return.
- $R^g(\pi_i) \leq R_{max}$ force the policy to train on goals that still need improvements.

Adversarial Goal Generation

Use of a GAN in order to sample new goals uniformly from GOID.

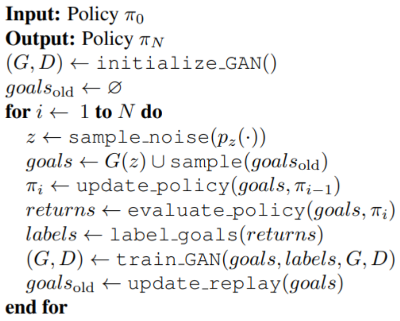
- Goal Generator neural network G(z) to generate goals from a noise vector z. Trained to uniformly output goals in GOID using a goal discriminator neural network.
- Goal Discriminator neural network D(g) trained to whether a goal is at the appropriate level of difficulty for the current policy (GOID).

$$\min_D V(D) = \mathbb{E}_{g\sim p_{\text{data}}(g)}\Big[y_g(D(g)-b)^2 + (1-y_g)(D(g)-a)^2\Big] + \mathbb{E}_{z\sim p_z(z)}[(D(G(z))-a)^2]$$

$$\min_G V(G) = \mathbb{E}_{z\sim p_z(z)}[D(G(z))-c)^2]$$

Policy Optimization

Use these new goals to efficiently train the policy, improving its coverage objective. **TRPO** with **GAE** is used as a reinforcement learning algorithm.



```
**Input:** Policy $\pi_0$
**Output:** Policy $\pi_N$
$(G, D) \leftarrow$ initialize_GAN()
$goals_{\text{old}} \leftarrow \varnothing$
**for** $i \leftarrow 1$ **to** $N$ **do**
    $z \leftarrow$ sample_noise$(p_z(\cdot))$
    $goals \leftarrow G(z) \cup$ sample$(goals_{\text{old}})$
    $\pi_i \leftarrow$ update_policy$(goals, \pi_{i-1})$
    $returns \leftarrow$ evaluate_policy$(goals, \pi_i)$
    $labels \leftarrow$ label_goals$(returns)$
    $(G, D) \leftarrow$ train_GAN$(goals, labels, G, D)$
    $goals_{\text{old}} \leftarrow$ update_replay$(goals)$
**end for**
```

HOW IT WORKS

Goal GAN Objective

Hyperparameters: label for **fake** data a=-1, label for **real** data b=1, value that the generator wants the discriminator to believe for fake data c=0.

Discriminator Value Function

$$\min_D V(D) = \mathbb{E}_{g\sim p_{data}(g)}\left[\boxed{y_g(D(g)-b)^2} + \boxed{(1-y_g)(D(g)-a)^2}\right] + \mathbb{E}_{z\sim p_z(z)}\left[\boxed{(D(G(z))-a)^2}\right]$$

when the sampled goal is a GOID, $y_g = 1$, the distance between the output of the discriminator and the goal label is minimized

Conversely $y_g = 0$, the distance with the label for non-GOID goals is minimized

Goal extracted from the goal set containing both goals in GOID and goals not in GOID

Sample a noise vector from a distribution of noise vectors

Generates the goal from the noise vector via the generator and trains the discriminator to recognize it as not GOID

$$\min_G V(G) = \mathbb{E}_{z\sim p_z(z)}[D(G(z)) - c)^2]$$

Generator Value Function

train the generator to uniformly output goals in GOID

Generative Goal Learning

```
Input: Policy $\pi_0$
Output: Policy $\pi_N$
$(G, D) \leftarrow$ initialize_GAN()
$goals_{old} \leftarrow \varnothing$
for $i \leftarrow 1$ to $N$ do
    $z \leftarrow$ sample_noise($p_z(\cdot)$)
    $goals \leftarrow G(z) \cup$ sample($goals_{old}$)
    $\pi_i \leftarrow$ update_policy($goals, \pi_{i-1}$)
    $returns \leftarrow$ evaluate_policy($goals, \pi_i$)
    $labels \leftarrow$ label_goals($returns$)
    $(G, D) \leftarrow$ train_GAN($goals, labels, G, D$)
    $goals_{old} \leftarrow$ update_replay($goals$)
end for
```

Generate a set of goals

Train the policy using **TRPO** with **GAE**

Use the new policy to label each goal label

Use new labels to train the goal generator and goal discriminator

Use replay buffer to prevent catastrophic forgetting

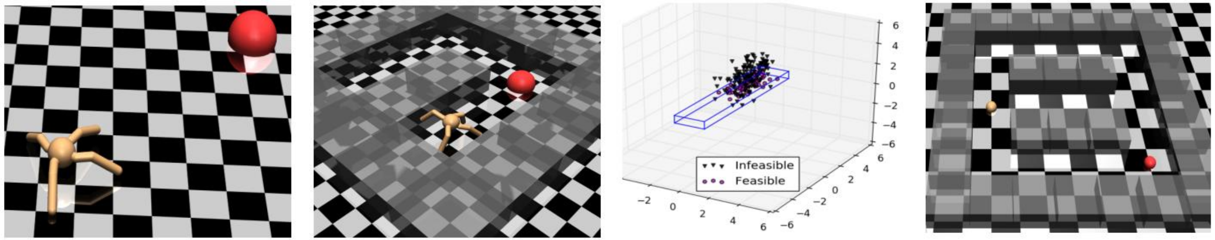
EXPERIMENTS

The method is demonstrated in challenging robotic locomotion tasks, where the **goals** are the (x, y) position of the Center of Mass (CoM) of a dynamically complex quadruped agent. The locomotion tasks are:

- The agent is inside an environment **without constraints** navigating in this free space.
- The agent is inside a U-Maze environment trained to **reach the other side of the U-turn**.
- Train a point-mass agent to reach any point within a multi-path maze.
- Scale to higher-dimensional goal-spaces with a low-dimensional spaces of feasible goals.

Goal GAN method is compared against four baselines.

- **Uniform Sampling** is a method that does not use a curriculum, training at every iteration on goals uniformly sampled from the goal-space.
- **Uniform Sampling with L2 loss** it receives the negative L2 distance to the goal as a reward at every step. It is tested to demonstrate that a straight-forward distance reward can be prone to local minima.
- Two methods adapted from the literature: **Asymmetric Self-play** method and **SAGGRIAC** method.
- An ablation (GAN fit all) and an oracle are also provided to the method to better understand the importance of sampling goals in GOID.



(a) Free Ant Locomotion (b) Maze Ant Locomotion (c) Point-mass 3D (d) Multi-path point-mass

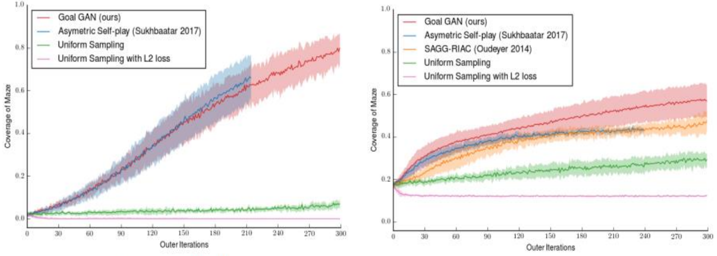
RESULTS

Free Ant and Maze Ant compared to baselines

It can be seen that the method leads to **faster training** compared to the baselines. The **Uniform Sampling** baseline does very poorly because too many samples are wasted attempting to train on goals that are infeasible for the current policy. With **L2 loss**, the agent falls into a poor local optima of not moving to avoid further negative rewards.

Asymmetric Self-play needs to train the goal-generating policy at every outer iteration, with an amount of rollouts equivalent to the ones used to train the goal-reaching policy, being therefore at least half as sample-efficient as the plots indicate**.**

SAGG-RIAC maintains an ever-growing partition of the goal-space that becomes more and more biased towards areas that already have more sub-regions, leading to reduced exploration and slowing down the expansion of the policy’s capabilities.



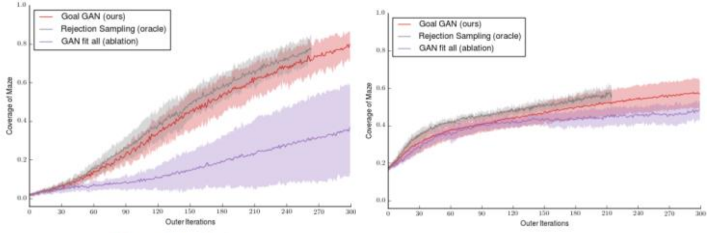
(a) Free Ant - Baselines

(b) Maze Ant - Baselines

Generating goals in GOID for efficient learning

The ablation of the method GAN fit all, that disregards the labels, performs worse because the expansion of the goals is not related to the current performance of the policy.

An upper bound on the performance is provided by using the Rejection Sampling to sample goals uniformly from GOID instead of Goal GAN. The performance of the method is quite close to the performance of the oracle.

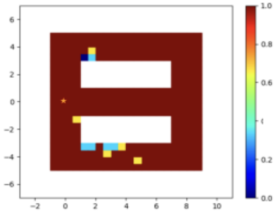


(c) Free Ant - Variants

(d) Maze Ant - Variants

Multi-path point-mass maze

It can be observed that the method produces a multi-modal distribution over goals, tracking all the areas where goals are at the appropriate level of difficulty. The star indicates the start position, each grid cell is colored according to the expected return achieved. Red indicates 100% of success.



Itr 100: Coverage=0.98

N-dimensional Point Mass

The **uniform sampling** baseline has poor performance as the number of dimensions increases because the fraction of feasible states within the full state space decreases as the dimension increases. The **GAN fit all** variation of the method suffers from the increase in dimension because it is not encouraged to track the narrow feasible region.

The **oracle** and the baseline with an **L2 distance reward** have perfect performance.

CONCLUSIONS & COMMENTS

A method for **automatic curriculum generation** that considerably improves the sample efficiency of learning to reach all feasible goals in the environment was presented. In this reinforcement learning paradigm the objective is to **train a single policy to succeed on a variety of goals**, under sparse rewards.

Generative adversarial training to automatically generate goals for the policy that are at the appropriate level of difficult was used.

Learning to reach **multiple goals** is useful for multi-task settings such as navigation or manipulation, in which we want the agent to perform a wide range of tasks.

However, the method can be improved by using better methods for generating goals since the percentage of generated goals that are at the appropriate level of difficulty (in GOID) stays around **20%**, and by using better ways to select the next goal to train on.